ELSEVIER Science of Computer Programming 44 (2002) 131–132

Science of Computer Programming

www.elsevier.com/locate/scico

Foreword

This special issue of Science of Computer Programming is devoted to applications of graph transformations in computer science. The research area of graph transformation dates back to the early seventies and started with mainly theoretical considerations. Meanwhile, graph transformation has become attractive as a modeling and programming paradigm for complex systems based on graph-like structures in a large variety of areas in computer science and related fields. During the Joint APPLIGRAPH/GETGRATS Workshop on Graph Transformation Systems (GRATRA 2000)—a satellite event of the European Joint Conferences on Theory and Practice of Software (ETAPS) 2000 in Berlin—35 lectures were presented by workshop participants from all over the world. GETRATS was a European network and APPLIGRAPH is a European working group running in the field of graph transformation systems. In particular, APPLIGRAPH is dedicated towards applications of graph transformation systems.

The participants who presented applications of graph transformations were invited to submit a full version of their presentation to Science of Computer Programming. After carefully refereeing all submissions, 5 papers have been accepted for this special issue which present applications of graph transformation in

- software architecture,
- visual languages, and
- agent-based systems.

In these papers, the concept of graph transformation is used to describe and manipulate the graph-like representation of software architectures, to define visual languages and to develop appropriate support tools, to define consistent transformations of visual languages into alternative representations for verification and validation purposes, and finally as a modeling means itself for describing agent-specific aspects in a software system.

In *A graph transformation approach to software architecture reconfiguration* Wermelinger and Fiadeiro propose a uniform algebraic approach to reconfigure software architectures in order to adapt them to new requirements or a changing environment. The approach assumes that components are written in a high-level program design language. Based on this, the approach deals with typical problems such as guaranteeing that new components are introduced in the correct state and that the resulting architecture conforms to certain structural constraints. The contribution improves previous work in the area of Architecture Description Languages (ADL) and related fields.

The next two contributions present approaches for using graph transformations as a formal means for defining visual languages and corresponding support tools.

0167-6423/02/$ - see front matter
PII: S0167-6423(02)00035-7

Minas presents *Concepts and realization of a diagram editor generator based on hypergraph transformation*. This paper describes DiaGen, a rapid prototyping tool for creating diagram editors which supports syntax-directed as well as free-hand editing. Created editors use hypergraphs as internal diagram models and hypergraph parsers for syntactic analysis. Syntax-directed editing is realized by programmed hypergraph transformation. This approach has proven to be powerful and general in the sense that it supports quick prototyping of diagram editors for a variety of languages.

In the same line of research Bardohl presents GenGEd—*A visual environment for visual languages*. GenGEd supports the visual definition of visual languages where each language is defined by an alphabet and a grammar. Based on a specific language definition a graphical editor is generated which allows the syntax-directed editing of diagrams over the specified language. The GenGEd approach is based on algebraic graph transformation and graphical constraint solving.

In the remaining two contributions of this special issue, graph transformation concepts are deployed to improve the modeling phase within a software development process.

In *Designing the automatic transformation of visual languages* Varro, Varro and Pataricza present a general framework for an automated model transformation system. This is used to verify and validate designed system models by deploying existing formal verification tools prior of the implementation in order to avoid costly re-design cycles. The authors use the UML (Unified Modeling Language) as system modeling language and show how mathematical models of various formal verification tools are automatically derived from UML diagrams by mathematical transformations. These transformations are based on graph transformations and planner algorithms of artificial intelligence.

In *Formal agent-oriented modeling with UML and graph transformation* Depke, Heckel and Küster present a dedicated modeling language for agent-based systems. The agent paradigm can be seen as an extension of the notion of (active) objects by concepts like autonomy and cooperation. Mainstream object-oriented modeling techniques do not account for these agent-specific aspects. In this contribution, graph transformation is used both on the level of modeling in order to capture agent-specific aspects and as underlying formal semantics of the approach. In particular, concepts of the concurrency theory of graph transformation are exploited in this agent-oriented modeling approach.

The five contributions of this special issue show that graph transformation concepts have evolved towards a mature means to be used in different areas of computer science. Graph transformations are particularly suited in all cases, where graph-like structures are to be formally defined and are to be consistently transformed into alternative representations.

We are grateful to all the referees of these papers, and to Egidio Astesiano and Elsevier Science for fruitful cooperation in editing this special issue.

Hartmut Ehrig, Gregor Engels*, Hans-Jörg Kreowski, Gabriele Taentzer
Guest Editors

*Contact person. *E-mail address:* engels@uni-paderborn.de.